Set-2

1. Extend the lexical Analyzer to Check comments, dened as follows in C:

a) A comment begins with // and includes all characters until the end of that line.

b) A comment begins with /* and includes all characters through the next occurrence of the character sequence */Develop a lexical Analyzer to identify whether a given line is a comment or not.

Code:

```
%{

#include <stdio.h>

%}

%%

"//".*      { printf("Single-line comment: %s\n", yytext); }

"/"([^]|\+[^/])("/")  { printf("Multi-line comment: %s\n", yytext); }

.*          { printf("Not a comment: %s\n", yytext); }

%%

int main() {

    printf("Enter a line of code: ");

    yylex();

    return 0;

}

int yywrap() {

    return 1;

}
```

2. Implement a C program to perform symbol table operations.

Code:

```
#include <stdio.h>

#include <string.h>

#include <stdlib.h>
```

```
#define MAX 100

typedef struct {
    char name[50];
    char type[10];
    int address;
} Symbol;

Symbol table[MAX];
int count = 0;

void insert(char *name, char *type, int address) {
    for (int i = 0; i < count; i++) {
        if (strcmp(table[i].name, name) == 0) {
            printf("Symbol already exists!\n");
            return;
        }
    }
    strcpy(table[count].name, name);
    strcpy(table[count].type, type);
    table[count].address = address;
    count++;
    printf("Symbol inserted successfully.\n");
}

void display() {
    printf("\nSymbol Table:\n");
    printf("----------------------------------\n");
    printf("Name\tType\tAddress\n");
    printf("----------------------------------\n");
    for (int i = 0; i < count; i++) {
        printf("%s\t%s\t%d\n", table[i].name, table[i].type, table[i].address);
    }
}

int search(char *name) {
    for (int i = 0; i < count; i++) {
        if (strcmp(table[i].name, name) == 0) {
            printf("Symbol found at address: %d\n", table[i].address);
            return i;
        }
    }
    printf("Symbol not found!\n");
    return -1;
}

int main() {
    int choice;
    char name[50], type[10];
    int address;

    while (1) {
        printf("\n1. Insert Symbol\n2. Display Symbol Table\n3. Search Symbol\n4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
    switch (choice) {
      case 1:
        printf("Enter name, type, and address: ");
        scanf("%s %s %d", name, type, &address);
        insert(name, type, address);
        break;
      case 2:
        display();
        break;
      case 3:
        printf("Enter name to search: ");
        scanf("%s", name);
        search(name);
        break;
      case 4:
        exit(0);
      default:
        printf("Invalid choice!\n");
    }
  }
  return 0;
}
```

Orr

```
#include <stdio.h>
#include <string.h>
#define MAX 100

struct Symbol {
    char name[50];
    char type[20];
    int address;
} table[MAX];
int count = 0;
void insert(char *name, char *type, int address) {
    strcpy(table[count].name, name);
    strcpy(table[count].type, type);
    table[count].address = address;
    count++;
    printf("Inserted: %s\n", name);
}
void display() {
    printf("\nSymbol Table:\n");
    printf("Name\tType\tAddress\n");
    for (int i = 0; i < count; i++) {
        printf("%s\t%s\t%d\n", table[i].name, table[i].type, table[i].address);
    }
}
int main() {
    insert("x      ", "int       ", 123);
    insert("y      ", "float     ", 104);
    insert("z      ", "char      ", 108);
    display();
    return 0;
}
```

3. Write a LEX program to recognize a word and relational operator.

Code:

```
%{
#include <stdio.h>
%}

%%

[a-zA-Z_][a-zA-Z0-9_]*    { printf("Word: %s\n", yytext); }
(==|!=|<=|>=|<|>)         { printf("Relational Operator: %s\n", yytext); }
[ \t\n]                   ; // Ignore whitespace
.                         { printf("Invalid Token: %s\n", yytext); }

%%

int main() {
   printf("Enter input: ");
   yylex();
   return 0;
}

int yywrap() {
   return 1;
}
```

4. Write a LEX program to count the number of Macros defined and header files included   in the C program.

Input Source Program: (sample.c)

```
#define PI 3.14
#include<stdio.h>
#include<conio.h>
```

```
 void main ()
{
int a,b,c = 30;
printf("hello");
}
```

Code:

```
%{
int nmacro, nheader;
%}
%%
^#define { nmacro++; }
^#include { nheader++; }
.|\n { }
%%
int yywrap(void) {
return 1;
}
int main(int argc, char *argv[]) {
yyin = fopen(argv[1], "r");
yylex();
printf("Number of macros defined = %d\n", nmacro);
printf("Number of header files included = %d\n", nheader);
fclose(yyin);
}
```